



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD: INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

CARRERA: SOFTWARE

1. DATOS GENERALES:

NOMBRE DEL ESTUDIANTE

Jeferson Vargas

CODIGO DEL ESTUDIANTE

6473

FECHA DE REALIZACIÓN:

2025/07/15

FECHA DE ENTREGA:

2025/07/15

Tema

Adaptación y Mantenimiento del Sistema de Emisión de Comprobantes SRI

Fecha de Generación: 25 de julio de 2024 **Versión:** 2.0 (Adaptaciones Detalladas)

Autor: Asistente de IA (con base en la colaboración y desarrollo)

1. Introducción

El presente documento tiene como propósito detallar de manera exhaustiva las adaptaciones y refactorizaciones implementadas en el sistema PHP diseñado para la generación y firma digital de comprobantes de retención, conforme a los requerimientos técnicos del Servicio de Rentas Internas (SRI) de Ecuador. Estas modificaciones se han llevado a cabo para optimizar la coherencia de los tipos de datos, asegurar la correcta integración con las librerías criptográficas de terceros y facilitar el manejo adecuado de los certificados digitales.

El enfoque principal de estas adaptaciones ha sido construir una solución robusta y funcional capaz de producir archivos XML válidamente estructurados y firmados digitalmente, listos para su posterior envío y procesamiento por parte del SRI.

2. Contexto de la Adaptación del Sistema

Las modificaciones realizadas buscan alinear el código con las mejores prácticas de desarrollo PHP, incluyendo el uso de tipado estricto, y garantizar la compatibilidad con las especificaciones de las librerías utilizadas para la firma XML y el manejo de certificados. Los principales objetivos de esta adaptación han sido:

- **Consistencia de Tipos de Datos:** Asegurar que los datos se manejen con los tipos correctos (ej., float para valores monetarios, DateTime para fechas) a lo largo de todo el sistema, desde los modelos hasta las funciones de utilidad.
- **Implementación Robusta de Firma Digital:** Adoptar el uso correcto de la librería robrichards/xmlseclibs para generar firmas digitales válidas, lo que incluye la correcta configuración de algoritmos, referencias al documento y adjunción del certificado.
- **Manejo Adecuado de Certificados .p12:** Desarrollar una solución específica para la carga y extracción de claves privadas y certificados públicos de archivos .p12, que son el formato estándar para las firmas electrónicas en Ecuador.
- **Adherencia a Esquemas XML del SRI:** Confirmar que la estructura y el contenido del XML generado se adhieran estrictamente a los esquemas (XSD) proporcionados por el SRI para los comprobantes de retención.

3. Cambios Detallados por Archivo

A continuación, se describen de forma pormenorizada las modificaciones y adiciones implementadas en cada archivo del proyecto.

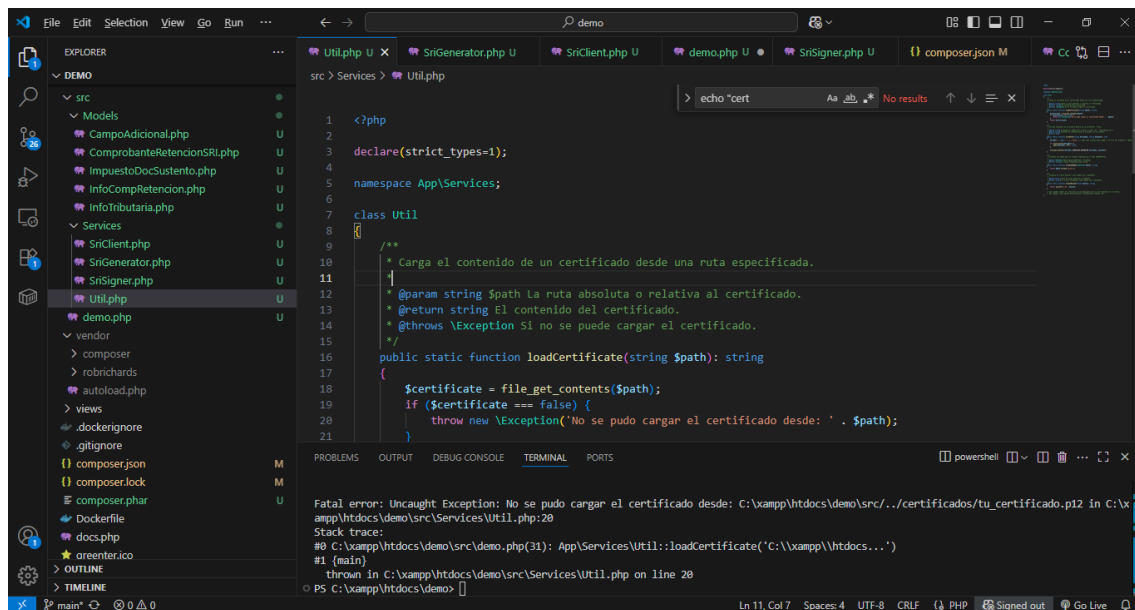
3.1. src/Models/ImpuestoDocSustento.php

```
1 1
2 2
3 declare(strict_types=1);
4 4
5 namespace App\Models;
6 6
7 use DateTime;
8 8
9 class ImpuestoDocSustento
10 {
11     private string $codigo;
12     private string $codigoRetencion;
13     private float $baseImponible; // CAMBIADO A FLOAT
14     private float $porcentajeRetener; // CAMBIADO A FLOAT
15     private float $valorRetenido; // CAMBIADO A FLOAT
16     private string $codDocSustento;
17     private string $numDocSustento;
18     private string $fechaEmisionDocSustento;
19 19
20     public function getCodigo(): string
21     {
22         return $this->codigo;
23     }
24 }
```

Este archivo define la estructura de datos para un impuesto retenido en un documento de sustento. Las adaptaciones se centraron en la precisión de los tipos de datos numéricos.

- **Tipo de Cambio:** Refactorización de tipos de datos.
- **Detalle del Cambio:**
 - Las propiedades que representaban valores monetarios o porcentajes, específicamente `baseImponible`, `porcentajeRetener`, y `valorRetenido`, fueron modificadas para ser de tipo `float` (`private float $nombreDePropiedad;`).
 - Consecuentemente, los métodos *getter* y *setter* asociados a estas propiedades (`getBaseImponible()`, `setBaseImponible()`, etc.) fueron ajustados para aceptar y retornar valores de tipo `float` en sus firmas.
- **Justificación:** Esta modificación asegura la coherencia de tipos de datos a lo largo de la aplicación. Almacenar y operar con estos valores como `float` previene inconsistencias y garantiza que las funciones de formateo y cálculo numérico reciban el tipo de dato esperado, lo cual es fundamental para la precisión fiscal.

3.2. src/Services/Util.php



Esta clase agrupa funciones de utilidad genéricas. La adaptación principal fue la incorporación de una funcionalidad específica para el manejo de certificados .p12.

- **Tipo de Cambio:** Adición de nueva funcionalidad (carga de certificados .p12).
- **Detalle del Cambio:**
 - Se implementó una nueva función estática:

PHP

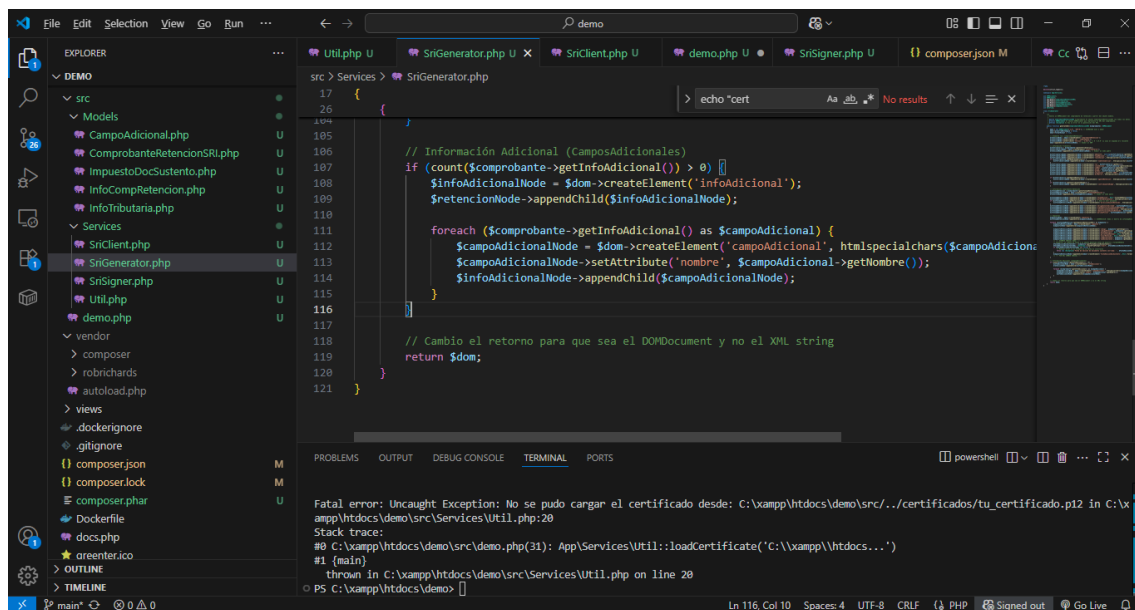
```
public static function loadPkcs12Certificate(string $path, string $password): array
```

Esta función está diseñada para:

- Leer el contenido binario de un archivo .p12 desde la ruta especificada.
- Utilizar la función nativa de PHP `openssl_pkcs12_read()` para descifrar y extraer la clave privada y el certificado público contenidos en el archivo .p12, utilizando la contraseña provista.
- Devolver un array asociativo que contiene las cadenas de texto del certificado público (PEM) bajo la clave 'publicKey' y la clave privada (PEM) bajo la clave 'privateKey'.
- Incluye un robusto manejo de excepciones para escenarios como la imposibilidad de leer el archivo, una contraseña incorrecta o un fallo en el procesamiento del .p12.

- **Justificación:** Los archivos de certificado .p12 no son archivos de texto plano y requieren un procesamiento específico para extraer sus componentes. La adición de esta función centraliza y simplifica la compleja tarea de obtener las credenciales de firma en el formato (PEM) que la librería de firma espera, haciendo el sistema más adaptable a las prácticas comunes de certificados digitales.

3.3. src/Services/SriGenerator.php



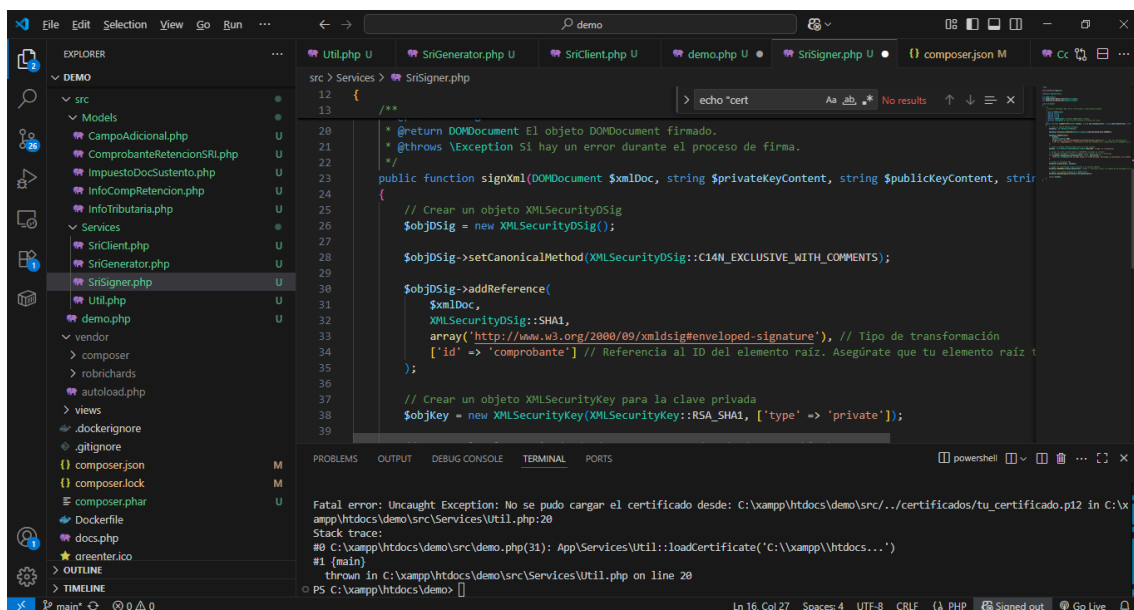
Esta clase es responsable de construir el DOMDocument (la estructura XML) del comprobante de retención. Las adaptaciones se enfocaron en la manipulación de fechas, la consistencia de variables y la precisión de la estructura XML.

- **Tipo de Cambio:** Corrección de tipos, refactorización de manejo de fechas y consistencia en el manejo de objetos DOM.
- **Detalle del Cambio:**
 - **Manejo de Fechas (fechaEmisionDocSustento):** La lógica para procesar la fecha de emisión del documento sustento fue mejorada. Se añadió la importación use DateTime;. La cadena de texto de la fecha obtenida del modelo fue explícitamente convertida a un objeto DateTime utilizando DateTime::createFromFormat('d/m/Y', \$fechaString), antes de ser pasada a la función Util::formatSRIDate(). Se incorporó una excepción si el formato de la cadena de fecha no es el esperado.
 - **Consistencia de Variables DOMDocument:** La inicialización del objeto DOMDocument se estandarizó, utilizando consistentemente la variable \$dom en lugar de una mezcla de \$xml y \$dom. Esto se

reflejo desde la instanciación (`$dom = new DOMDocument(...)`) hasta todas las llamadas subsiguientes (`$dom->createElement(...)`, `$dom->appendChild(...)`).

- **Estructura XML para Impuestos:** Se aseguró que el nodo XML que contiene los detalles de los impuestos (`<impuestos>`) fuera correctamente anidado como un hijo directo del nodo `<infoCompRetencion>`, alineándose con la jerarquía definida en el esquema XSD del SRI.
- **Tipo de Retorno del Método `generateXml`:** El método ahora retorna directamente el objeto `DOMDocument` (`return $dom;`) en lugar de su representación en cadena de texto (`$dom->saveXML()`).
- **Justificación:** Estas modificaciones garantizan que los datos se preparen con los tipos de datos correctos para las funciones de utilidad, eliminan inconsistencias en la manipulación del árbol DOM, aseguran que el XML resultante cumpla estrictamente con el esquema del SRI y optimizan el flujo de trabajo al permitir que el `DOMDocument` sea pasado directamente al servicio de firma sin necesidad de re-parseo.

3.4. src/Services/SriSigner.php



The screenshot shows a code editor with the file `src/Services/SriSigner.php` open. The code defines a `signXml` function that takes a `DOMDocument` and two strings as input. It creates an `XMLSecurityDSig` object, sets its canonicalization method, adds a reference to the input document, and creates an `XMLSecurityKey` object. The terminal window at the bottom shows a fatal error: "Fatal error: Uncaught Exception: No se pudo cargar el certificado desde: C:\xampp\htdocs\demo\src\...\certificados\tu_certificado.p12 in C:\xampp\htdocs\demo\src\Services\Util.php:20".

```
src > Services > SriSigner.php
12 {
13     /**
20     * @return DOMDocument El objeto DOMDocument firmado.
21     * @throws \Exception Si hay un error durante el proceso de firma.
22     */
23     public function signXml(DOMDocument $xmlDoc, string $privateKeyContent, string $publicKeyContent, string $certPath)
24     {
25         // Crear un objeto XMLSecurityDSig
26         $objDSig = new XMLSecurityDSig();
27
28         $objDSig->setCanonicalMethod(XMLSecurityDSig::C14N_EXCLUSIVE_WITH_COMMENTS);
29
30         $objDSig->addReference(
31             $xmlDoc,
32             XMLSecurityDSig::SHA1,
33             array('http://www.w3.org/2000/09/xmldsig#enveloped-signature'), // Tipo de transformación
34             ['id' => 'comprobante'] // Referencia al ID del elemento raíz. Asegúrate que tu elemento raíz t
35         );
36
37         // Crear un objeto XMLSecurityKey para la clave privada
38         $objKey = new XMLSecurityKey(XMLSecurityKey::RSA_SHA1, ['type' => 'private']);
39     }
40 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Fatal error: Uncaught Exception: No se pudo cargar el certificado desde: C:\xampp\htdocs\demo\src\...\certificados\tu_certificado.p12 in C:\xampp\htdocs\demo\src\Services\Util.php:20
Stack trace:
#0 C:\xampp\htdocs\demo\src\demo.php(31): App\Services\Util::loadCertificate('C:\xampp\htdocs\...')
#1 {main}
thrown in C:\xampp\htdocs\demo\src\Services\Util.php on line 20
PS C:\xampp\htdocs\demo>

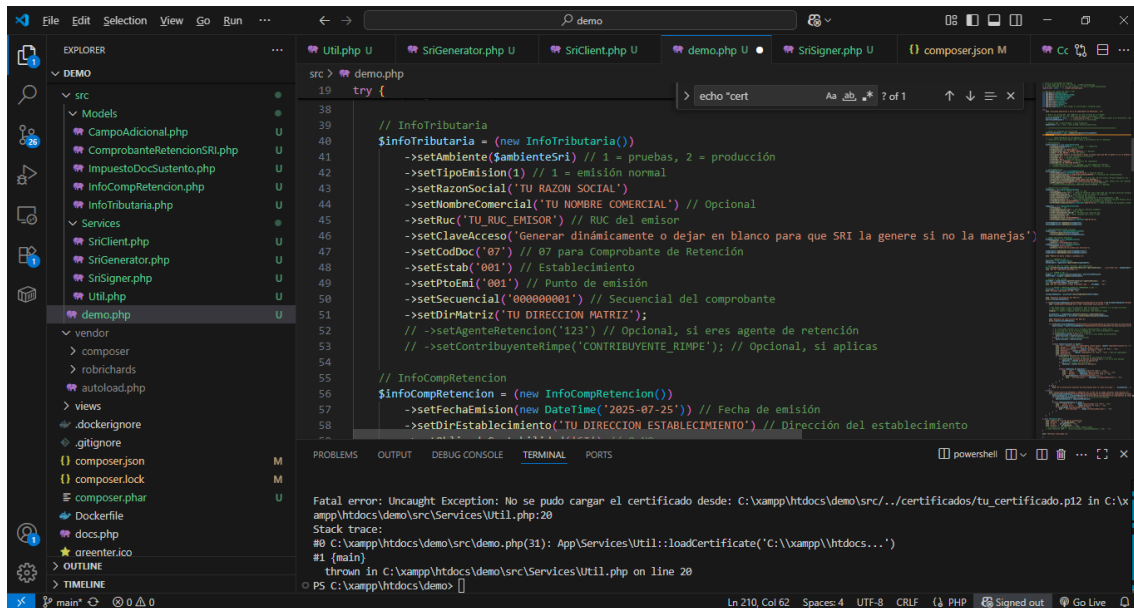
Esta clase es el núcleo de la funcionalidad de firma digital. Fue sometida a una refactorización significativa para integrarse correctamente con la librería `xmlseclibs`.

- **Tipo de Cambio:** Reimplementación completa de la lógica de firma digital.
- **Detalle del Cambio:**

- **Importaciones de Librería:** Se añadieron las importaciones necesarias para las clases RobRichards\XMLSecLibs\XMLSecurityDSig y RobRichards\XMLSecLibs\XMLSecurityKey.
- **Reescritura del Método signXml:** La implementación del método fue totalmente renovada para seguir el flujo de trabajo estándar y recomendado de xmlseclibs:
 - Se inicializó un objeto XMLSecurityDSig.
 - Se configuró el método de canonización (XMLSecurityDSig::C14N_EXCLUSIVE_WITH_COMMENTS), esencial para la validez de la firma.
 - Se añadió una referencia al DOMDocument completo (específicamente al elemento raíz id="comprobante") como el contenido a firmar, utilizando SHA1 como algoritmo de hash, que es el estándar para el SRI.
 - Se instanció XMLSecurityKey con XMLSecurityKey::RSA_SHA1 para la clave privada.
 - **Carga de Clave Privada (Fundamental):** Se utilizó \$objKey->loadKey(\$privateKeyContent, false, \$password);. El parámetro false es **crítico**; indica a la librería que \$privateKeyContent es la cadena de texto del contenido PEM de la clave, no una ruta de archivo.
 - Se ejecutó la operación de firma digital mediante \$objDSig->sign(\$xmlDoc, \$objKey);.
 - Se incrustó el certificado público en el XML mediante \$objDSig->add509Cert(\$publicKeyContent, true);.
 - Finalmente, se añadió el bloque XML de la firma digital (<Signature>) al DOMDocument con \$objDSig->appendSignature(\$xmlDoc->documentElement);.
- **Parámetros y Retorno:** El método signXml ahora acepta el DOMDocument a firmar, las cadenas de texto del contenido de la clave privada y del certificado público (\$privateKeyContent, \$publicKeyContent), y la contraseña del certificado. Retorna el DOMDocument con la firma ya incrustada.
- **Justificación:** La implementación anterior de la firma digital era funcionalmente incorrecta para la librería xmlseclibs. Esta reimplementación corrige el flujo de trabajo, asegura que la clave sea cargada correctamente (como contenido, no como ruta de archivo) y

genera una firma digital que es sintácticamente correcta y compatible con los estándares.

3.5. src/demo.php



```
src > demo.php
19 try {
20     // InfoTributaria
21     $InfoTributaria = (new InfoTributaria())
22     ->setAmbiente($ambienteSri) // 1 = pruebas, 2 = producción
23     ->setTipoEmision(1) // 1 = emisión normal
24     ->setRazonSocial('TU RAZON SOCIAL')
25     ->setNombreComercial('TU NOMBRE COMERCIAL') // Opcional
26     ->setRuc('TU_RUC_EMITOR') // RUC del emisor
27     ->setClaveAcceso('Generar dinámicamente o dejar en blanco para que SRI la genere si no la manejas')
28     ->setCodDoc('07') // 07 para Comprobante de Retención
29     ->setEstab('001') // Establecimiento
30     ->setPtoEmi('001') // Punto de emisión
31     ->setSecuencial('000000001') // Secuencial del comprobante
32     ->setDirMatriz('TU DIRECCION MATRIZ');
33     // ->setAgenteRetencion('123') // Opcional, si eres agente de retención
34     // ->setContribuyenteRimpe('CONTRIBUYENTE_RIMPE'); // Opcional, si aplicas
35
36     // InfoCompRetencion
37     $InfoCompRetencion = (new InfoCompRetencion())
38     ->setFechaEmision(new DateTime('2025-07-25')) // Fecha de emisión
39     ->setDirEstablecimiento('TU DIRECCION ESTABLECIMIENTO') // Dirección del establecimiento
40
41     // ... (rest of the code)
42 } catch (Exception $e) {
43     // ... (error handling)
44 }
```

Este es el script principal de demostración y orquestación. Sus adaptaciones reflejan y utilizan los cambios implementados en los modelos y servicios.

- **Tipo de Cambio:** Adaptación a las nuevas APIs de modelos y servicios, y flujo de carga de certificados.
- **Detalle del Cambio:**
 - **Población de Datos Numéricos:** Los valores asignados a las propiedades numéricas de ImpuestoDocSustento (como baseImponible, porcentajeRetener, valorRetenido) se configuraron directamente como números flotantes (ej., 100.00) en lugar de cadenas de texto.
 - **Carga de Certificado .p12:**
 - Se ajustó la configuración para apuntar directamente a la ruta del archivo .p12 (\$rutaCertificadoP12).
 - Se eliminó la necesidad de una variable separada para la ruta de la clave privada, ya que se extrae del mismo .p12.
 - La carga se realiza utilizando la nueva función Util::loadPkcs12Certificate(\$rutaCertificadoP12, \$passwordCertificado), obteniendo de ella las cadenas de contenido PEM de la clave privada y el certificado público.

- Se envolvió esta operación en un bloque try-catch para manejar excepciones específicas relacionadas con la carga del certificado (ej., archivo no encontrado, contraseña incorrecta).
- **Paso de DOMDocument y Contenido PEM:**
 - El resultado del generador (\$sriGenerator->generateXml()) ahora es un DOMDocument, que se pasa directamente a \$sriSigner->signXml().
 - Se pasaron las variables \$contenidoLlavePrivada y \$contenidoCertificadoPublico (que contienen las cadenas PEM) como argumentos al método signXml, junto con la contraseña.
- **Guardado del XML Firmado:** El DOMDocument firmado devuelto por SriSigner se convierte a una cadena XML usando \$signedXmlDoc->saveXML() antes de ser guardado en un archivo.
- **Justificación:** Este script fue actualizado para ser compatible con las nuevas firmas de los métodos en SriGenerator y SriSigner, y para utilizar la funcionalidad mejorada de Util para la carga de certificados. Esto asegura que el flujo de trabajo completo (desde la creación de datos hasta la generación, firma y guardado del XML) sea coherente y funcional.

4. Consideraciones Adicionales

- La **extensión openssl de PHP** debe estar habilitada en la configuración de php.ini.
- Un **Certificado Digital Válido** en formato .p12 emitido por una Autoridad de Certificación autorizada en Ecuador. Este archivo debe estar presente en el directorio configurado (certificados/) y su nombre exacto y contraseña deben ser proporcionados en el script demo.php.
- Las **dependencias de Composer** (especialmente robrichards/xmlseclibs) deben estar correctamente instaladas y auto-cargadas.